

Programação Sistema em C

Revisão de IAED

Exercício 1

- `int *x; *x = 100;`
- Qual o problema? Como o resolver?
- Soluções:
`int y; x=&y;`
ou
`x=(int*)malloc(sizeof(int));`

Exercício 2

- `char *s = "texto inicial";`
- Qual o problema? Como o resolver?
- Nenhum, pois a declaração aloca implicitamente memória para a string indicada.

Exercício 3

```
char t1[] = "ola";
char *t2 = "ola";
```

- Qual a semelhança entre as duas declarações?
- Ambas alocam 4 bytes e copiam para essa posição de memória a sequência de caracteres 'o','l','a','\0'
- Em ambos os casos é possível modificar o conteúdo da memória alocada
- É permitido fazer
`t2=t1;`
Mas não é permitido fazer
`t1=t2;`

Exercício 4

- ```
int *x = (int*)malloc(sizeof(int));
...
x=NULL;
```
- Qual o problema? Como resolver?
- Falta libertar a memória alocada (caso nenhum outro ponteiro referencie o apontado por x).
- Solução: `free(x);`

## Exercício 5

- ```
typedef struct {
    int *x;
    char *y;
} mystruct;
```



```
mystruct *p = (mystruct*)malloc(sizeof(mystruct));
*(p->x)=3;
```
- Qual o problema? Como o resolver?
- Apenas alocámos memória para a estrutura.
- Faltou
`p->x = (int*)malloc(sizeof(int));`
`p->y = (int*)malloc(sizeof(int));`

Exercício 6

- mystruct *novaEstrutura() {
 mystruct s;
 s.x=NULL; s.y=NULL;
 return &s;
}
- Qual o problema? Como resolver?
- Retorno aponta para variável local da função, que deixará de existir quando a função retorna!
mystruct *s = (mystruct *)malloc(sizeof(mystruct));
s->x=NULL; s->y=NULL;
return s;

Se não responderam certo a todas as questões, então precisam rever a matéria de IAED!

- Consultar slides no site de IAED.
- Tirar dúvidas nos horários de dúvidas de SO.

Chamadas a Funções Sistema

- Usar comando *man* para documentação sobre cada função
- Exemplo: *man pipe*

pipe(2) - Linux man page

Name

pipe - create pipe

Synopsis

`#include <unistd.h>`

`int pipe(int filedes[2]);`

Description

`pipe()` creates a pair of file descriptors, pointing to a pipe inode, and places them in the array pointed to by `filedes`. `filedes[0]` is for reading, `filedes[1]` is for writing.

Return Value

On success, zero is returned. On error, -1 is returned, and `errno` is set appropriately.

Errors

`EINVAL` `filedes` is not valid. `EMFILE` Too many file descriptors are in use by the process. `ENFILE` The system limit on the total number of open files has been reached.

Necessário ter estes #include no nosso programa

Assegurar que os tipos de parâmetros são os mesmos no programa chamador

Atenção a todos os valores de retorno possíveis

No caso de erro, algumas funções indicam mais detalhe no inteiro `errno`

Algumas funções precisam também de opções adicionais de compilação/linkagem
(Essas situações são indicadas também na man page.)

Exercício

- Implemente a função `void dorme(int s)`, que adormece o processo durante `s` segundos (aproximadamente).

- *man sleep*

sleep(3) - Linux man page

Name

sleep - Sleep for the specified number of seconds

Synopsis

`#include <unistd.h>`

`unsigned int sleep(unsigned int seconds);`

Description

`sleep()` makes the current process sleep until `seconds` seconds have elapsed or a signal arrives which is not ignored.

Return Value

Zero if the requested time has elapsed, or the number of seconds left to sleep.